2020

FlexisourceIT Django Examination

QUITLONG.Joanna Marie

[TRADING SYSTEM: API MANUAL]

The document manual provides quick overview on the components of the trading system REST API as well as all possible endpoints that end user can use or access.

Table of Contents

List of Tables

1 Introduction

The application caters the development of REST API of a Trading System. The primary and required endpoints are authentication, placing trade and retrieving total value invested in a single stock. With this, it is designed to have three (3) primary models: User, Stock and StockOrder.

1. 1 User

The application used the default model user from the Django; possible fields along with the corresponding description are defined in the *Table 1 User Fields*. We must note also that there are still more available fields, listed are the chosen field that is indicated also in the serializer.

Field	Is Required?	Description
id	No	Id of the user in the database, no need to pass. It is auto increment and unique field.
first_name	No	Given name of the user
last_name	No	Last name of the user
email	No	Email contact of the user
username	Yes	Credential for user login
password	Yes	Credential for user login

Table 1 User Fields

It is decided to use a User model since as indicate in the requirement, "*we want to allow authenticated users…*". So with this, aside from super user, we can create our very own user as well as the basic CRUD (Create Read Update Delete) operations.

1.2 Stock

In the stock model, since we have to buy/sell a stock, of course we have to have a CRUD for the stock. The possible fields along with the corresponding description are defined in *Table 2 Stock Fields*.

Field	Is Required?	Description
id	No	Id of the stock in the database, no need to pass. It is auto increment and unique field.
name	Yes	Label or classification of stock. Maximum of 200 characters
price	No	Price of the stock. If not stated, default is 0.0.

Table 2 Stock Fields

1.3 StockOrder

In the stock order, it holds all the transaction made by the logged in user (either buy or sell of stock). The possible fields along with the corresponding description are defined in Table 3 StockOrder Fields.

Field	Is Required?	Description
id	No	Id of the stock order in the database, no need to pass. It is auto increment and unique field.
stock	No	Foreign key, related to the Stock model to have the stock information. This is automatically handled by Django
owner	No	Foreign key, related to the User model to have the user information. This is automatically handled by Django
quantity	Yes	Total number of stock that the logged in user wants to buy or sell.
created_time	No	Time that the logged in user buys or sells a stock or the time that the user made the transaction.

Table 3 StockOrder Fields

2 Available Endpoints

We have discussed the created and used models, we can proceed on the discussion on the available API endpoints and how the user can use it.

2.1 User Account

2.1.1 Create Account [POST] /api/v1/account/

- **Description:** Creation of new account for trading system
- **Request Method :** POST
- **Request Headers**

KEY	VALUE
Authorization	Token <YOUR_TOKEN>

- **Request Body**

```
{
    "username": "entered_username",
    "password": "entered_password",
    "first_name": "entered_first_name",
    "last_name": "entered_last_name",
    "email": "entered_email@gmail.com"
}
```

- **Response**

  Code 201

```
{
    "id": 9,
    "username": "entered_username",
    "first_name": "entered_first_name",
    "last_name": "entered_last_name",
    "email": "entered_email@gmail.com"
}
```

2.1.2 Update Account [PUT] /api/v1/account/{user_id}/

- **Description**: Update existing account in trading system
- **Request Method :** PUT
- **Request Headers**

KEY	VALUE
Authorization	Token <YOUR_TOKEN>

- **Request Body**

```
{
    "username": "new_username",
    "password": "new_password",
    "first_name": "new_first_name",
    "last_name": "new_last_name",
    "email": "new_email@gmail.com"
}
```


- **Response**
  Code 200


```
{
    "id": 9,
    "username": "new_username",
    "first_name": "new_first_name",
    "last_name": "new_last_name",
    "email": "new_email@gmail.com"
}
```


2.1.3 Sign In Account [POST] /api/v1/signin

- **Definition:** Allow user to sign in and to get their corresponding session token
- **Request Method :** POST
- **Request Headers:** N/A
- **Request Body**


```
{
    "username": "username",
    "password": "password"
}
```


- **Response**
  Code 200


```
{
    "user": {
        "id": 10,
        "username": "username",
        "first_name": "First Name",
        "last_name": "Last Name",
        "email": "email@email.com"
    },
    "expires_in": "86399.908028",
    "token": "91a80a7d4762eb6741759c06287043f0ee2ffd9d"
}
```

2.2 Stock

2.2.1 Create New Stock Item [POST] /api/v1/stock/

- **Definition:** Create new stock item that the user can buy or sell
- **Request Method :** POST
- **Request Headers**

KEY	VALUE
Authorization	Token <YOUR_TOKEN>

- **Request Body**

```
{
"name": "stock_name",
"price": 10
}
```

- **Response**

  Code 201

```
{
    "id": 4,
    "name": "stock_name",
    "price": 10.0
}
```

2.2.2 Update Existing Stock Item [PUT] /api/v1/stock/{stock_id}/

- **Definition:** Update existing stock that the user can buy or sell
- **Request Method :** PUT
- **Request Headers**

KEY	VALUE
Authorization	Token <YOUR_TOKEN>

- **Request Body**

```
{
"name": "stock_name",
"price": 10
}
```

- **Response**

  Code 200

```
{
    "id": 4,
    "name": "stock_name",
    "price": 10.0
}
```


2.2.3 Delete Existing Stock Item [DELETE] /api/v1/stock/{stock_id}/

- **Definition:** Delete existing stock record
- **Request Method :** DELETE
- **Request Headers**

KEY	VALUE
Authorization	Token <YOUR_TOKEN>

- **Request Body:** N/A
- **Response:** Code 200

2.2.4 Place Trade [POST] /api/v1/stock/{stock_id}/place-trade/

- **Definition:** User can buy or sell a stock. Indicate positive values if buy while negative values if sell
- **Request Method :** POST
- **Request Headers**

KEY	VALUE
Authorization	Token <YOUR_TOKEN>

- **Request Body**


```
{
    "quantity": 1
}
```


- **Response**

  Code 200


```
{
    "id": 13,
    "owner": 4,
    "quantity": 1,
    "created_time": "2020-08-26T01:24:23.053825Z"
}
```

2.2.5 Retrieve Total Investment [GET] /api/v1/stock/{stock_id}/total-invested/

- **Definition:** User can retrieve the total invested in a single stock. To calculate, we fetch the total order placed to a single stock then multiply it to the price of the stock. Take note, we have positive and negative quantity just to denote the buy and sell scheme but the minimum is 0 because we are tracing if the user has enough stock to sell a stock.
- **Request Method :** GET
- **Request Headers**

KEY	VALUE
Authorization	Token <YOUR_TOKEN>

- **Request Body:** NA
- **Response**

  Code 200


```
{
    "total_invested": 200.0
}
```